POC – Terraform

Summary:

Create a POC using different AWS Services like: IAM, S3, Lambda, API Gateway & CloudFront.

1. S3 (Simple Storage Service)

I.Brief:

An object storage service that offers industry-leading scalability, data availability, security, and performance

II.**Code** (aws_s3.tf, variable.tf, terraform.tfvars)

aws_s3.tf

```
resource "aws_s3_bucket" "api_bucket" {
  bucket = "${var.myregion}-${var.bucket_name}"
  tags = {
    Name        = "${var.myregion}-${var.bucket_name}"
    Environment = "${var.environment}"
  }
}
resource "aws_s3_bucket_acl" "api_acl" {
  bucket = aws_s3_bucket.api_bucket.id
  acl    = var.acl
  depends_on = [
    aws_s3_bucket.api_bucket
  ]
}
```

variable.tf

```
variable "bucket_name" {
  default = ""
}
variable "environment" {
  default = ""
}
variable "myregion" {
  default = ""
}
variable "acl" {
  default = ""
}
```

terraform.tf.vars

```
bucket_name       = "lambdabucketapi"
myregion          = "us-east-1"
environment       = "Dev"
acl               = "public-read-write"
```

III.**Code** (aws_s3.tf, variable.tf, terraform.tfvars) – Object Upload

aws_s3.tf

```
resource "aws_s3_object" "object" {
  bucket = "${var.myregion}-${var.bucket_name}"
  key    = var.object_key
  source = var.object_source
  acl = var.object_acl
  etag = filemd5(var.object_source)
  depends_on = [
    aws_s3_bucket.api_bucket
  ]
}
```

variable.tf

```
variable "object_key" {
  default = ""
}
variable "object_source" {
  default = ""
}
variable "object_acl" {
  default = ""
}
```

terraform.tf.vars

```
object_key          = "sample.txt"
object_source       = "Object/sample.txt"
object_acl          = "public-read-write"
```

IV.Implementation Steps:

Step:1

terraform init

```
PS C:\Users\kanchan.soni\Desktop\terraform blog> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v4.30.0...
- Installed hashicorp/aws v4.30.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Step:2

terraform plan:

```
PS C:\Users\kanchan.soni\Desktop\terraform blog> terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_s3_bucket.api_bucket will be created
  + resource "aws_s3_bucket" "api_bucket" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "-"
      + bucket_domain_name          = (known after apply)
      + bucket_regional_domain_name = (known after apply)
```

Step:3

terraform apply:

```
PS C:\Users\kanchan.soni\Desktop\terraform blog> terraform apply

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_s3_bucket.api_bucket will be created
  + resource "aws_s3_bucket" "api_bucket" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "-"
      + bucket_domain_name          = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = (known after apply)
      + policy                      = (known after apply)
      + region                      = (known after apply)
```

```
Plan: 3 to add, 0 to change, 0 to destroy.
aws_s3_bucket.api_bucket: Creating...
aws_s3_bucket.api_bucket: Creation complete after 7s [id=us-east-1-lambdabucketapi]
aws_s3_bucket_acl.api_acl: Creating...
aws_s3_object.object: Creating...
aws_s3_bucket_acl.api_acl: Creation complete after 1s [id=us-east-1-lambdabucketapi,public-read-write]
aws_s3_object.object: Creation complete after 2s [id=sample.txt]
```

2. Lambda

I.Brief:

Lambda is a serverless compute service that lets you build applications that respond quickly to new information and events

Reference: https://faun.pub/building-s3-event-triggers-for-aws-lambda-using-terraform-1d61a05b4c97

II.Services used as mentioned below:

aws_iam_role

aws_iam_role_policy

aws_lambda_function

aws_s3_bucket_notification

aws_lambda_permission

III.Code (lambda-function.tf, lambda-role.tf, variable.tf, terraform.tfvars, iam_files (json-files))

GitHub: https://github.com/KanchanSoni16/terraform-s3-lambda-apigatway.git

IV.Implementation Steps:

Step:1

terraform init

```
PS C:\Users\kanchan.soni\Desktop\terraform-blog> terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/archive from the dependency lock file
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/archive v2.2.0
- Using previously-installed hashicorp/aws v4.30.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Step:2

terraform plan:

```
PS C:\Users\kanchan.soni\Desktop\terraform-blog> terraform plan
data.archive_file.lambda_api_function: Reading...
data.archive_file.lambda_api_function: Read complete after 0s [id=b027074b33f87114baad1cc12d6a138cb13b47bc]
aws_s3_bucket.api_bucket: Refreshing state... [id=us-east-1-lambdabucketapi]
aws_s3_object.object: Refreshing state... [id=sample.txt]
aws_s3_bucket_acl.api_acl: Refreshing state... [id=us-east-1-lambdabucketapi,public-read-write]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # aws_iam_role.api_role will be created
  + resource "aws_iam_role" "api_role" {
      + arn                   = (known after apply)
      + assume_role_policy    = jsonencode(
            {
              + Statement = [
                  + {
                      + Action    = "sts:AssumeRole"
```

Step:3

terraform apply:

```
PS C:\Users\kanchan.soni\Desktop\terraform-blog> terraform apply
data.archive_file.lambda_api_function: Reading...
data.archive_file.lambda_api_function: Read complete after 0s [id=b027074b33f87114baad1cc12d6a138cb13b47bc]
aws_s3_bucket.api_bucket: Refreshing state... [id=us-east-1-lambdabucketapi]
aws_s3_object.object: Refreshing state... [id=sample.txt]
aws_s3_bucket_acl.api_acl: Refreshing state... [id=us-east-1-lambdabucketapi,public-read-write]

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:
```

```
aws_iam_role.api_role: Creating...
aws_iam_role.api_role: Creation complete after 3s [id=api_role]
aws_iam_role_policy.api_policy: Creating...
aws_lambda_function.AWS_S3_API_Lambda_Function: Creating...
aws_iam_role_policy.api_policy: Creation complete after 1s [id=api_role:api_policy]
aws_lambda_function.AWS_S3_API_Lambda_Function: Still creating... [10s elapsed]
aws_lambda_function.AWS_S3_API_Lambda_Function: Creation complete after 16s [id=AWS_S3_API_Lambda_Function]
aws_lambda_permission.allow_bucket: Creating...
aws_lambda_permission.allow_bucket: Creation complete after 1s [id=AllowS3Invoke]
aws_s3_bucket_notification.aws-lambda-trigger: Creating...
aws_s3_bucket_notification.aws-lambda-trigger: Creation complete after 2s [id=us-east-1-lambdabucketapi]
```

3. API Gateway

I.Brief:

An API gateway is an API management tool that sits between a client and a collection of backend services. An API gateway acts as a reverse proxy to accept all application programming interface (API) calls, aggregate the various services required to fulfil them, and return the appropriate result.

II.Services used as mentioned below:

aws_iam_role

aws_iam_role_policy

aws_api_gateway_rest_api

aws_api_gateway_resource

aws_api_gateway_method

aws_api_gateway_integration

aws_api_gateway_method_response

aws_api_gateway_integration_response

aws_lambda_permission

aws_api_gateway_deployment

III.Code (api_gateway.tf, apigatway-integration-role.tf, variable.tf, terraform.tfvars, iam_files (json-files))

GitHub: https://github.com/KanchanSoni16/terraform-s3-lambda-apigatway.git

IV.Implementation Steps:

Step:1

terraform init

```
PS C:\terraform\lambda> terraform init
Error opening log file: open C:\terraform\AWS_Infra\log: is a directory

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/archive from the dependency lock file
- Using previously-installed hashicorp/aws v4.30.0
- Using previously-installed hashicorp/archive v2.2.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Step:2

terraform plan:

```
PS C:\terraform\lambda> terraform plan
Error opening log file: open C:\terraform\AWS_Infra\log: is a directory
data.archive_file.lambda_api_function: Reading...
data.archive_file.lambda_api_function: Read complete after 0s [id=b027074b33f87114baad1cc12d6a138cb13b47bc]
aws_dynamodb_table.state-locking: Refreshing state... [id=state-locking]
aws_iam_role.api_role: Refreshing state... [id=api_role]
aws_s3_bucket.api_bucket: Refreshing state... [id=us-east-1-lambdabucketapi]
aws_s3_bucket.statefile-bucket: Refreshing state... [id=statefile-bucket]
aws_api_gateway_rest_api.rest_api: Refreshing state... [id=510utbv6w9]
aws_iam_role.apigatway_role: Refreshing state... [id=apigatway-role]
aws_iam_role_policy.apigatway_policy: Refreshing state... [id=apigatway-role:apigatway-policy]
aws_s3_bucket_acl.state_bucket_acl: Refreshing state... [id=statefile-bucket,private]
aws_api_gateway_resource.rest_api_resources: Refreshing state... [id=cqppoa]
aws_iam_role_policy.api_policy: Refreshing state... [id=api_role:api_policy]
aws_lambda_function.AWS_S3_API_Lambda_Function: Refreshing state... [id=AWS_S3_API_Lambda_Function]
aws_api_gateway_method.rest_api_method1: Refreshing state... [id=agm-510utbv6w9-cqppoa-GET]
aws_api_gateway_method_response.response_200: Refreshing state... [id=agmr-510utbv6w9-cqppoa-GET-200]
aws_api_gateway_integration.rest_api_integration1: Refreshing state... [id=agi-510utbv6w9-cqppoa-GET]
aws_lambda_permission.apigw_lambda: Refreshing state... [id=AllowExecutionFromAPIGateway]
aws_lambda_permission.allow_bucket: Refreshing state... [id=AllowS3Invoke]
aws_s3_object.object: Refreshing state... [id=sample.txt]
aws_s3_bucket_acl.api_acl: Refreshing state... [id=us-east-1-lambdabucketapi,public-read-write]
aws_api_gateway_integration_response.rest_api_integration_response: Refreshing state... [id=agir-510utbv6w9-cqppoa-GET-200]
aws_s3_bucket_notification.aws-lambda-trigger: Refreshing state... [id=us-east-1-lambdabucketapi]
aws_api_gateway_deployment.MyDemoDeployment: Refreshing state... [id=9i0tzx]
```

Step:3

terraform apply:

```
PS C:\terraform\lambda> terraform apply -auto-approve
Error opening log file: open C:\terraform\AWS_Infra\log: is a directory
data.archive_file.lambda_api_function: Reading...
data.archive_file.lambda_api_function: Read complete after 0s [id=b027074b33f87114baad1cc12d6a138cb13b47bc]
aws_dynamodb_table.state-locking: Refreshing state... [id=state-locking]
aws_iam_role.api_role: Refreshing state... [id=api_role]
aws_iam_role.apigatway_role: Refreshing state... [id=apigatway-role]
aws_s3_bucket.statefile-bucket: Refreshing state... [id=statefile-bucket]
aws_api_gateway_rest_api.rest_api: Refreshing state... [id=8ykbc3cw11]
aws_s3_bucket.api_bucket: Refreshing state... [id=us-east-1-lambdabucketapi]
aws_iam_role_policy.api_policy: Refreshing state... [id=api_role:api_policy]
aws_lambda_function.AWS_S3_API_Lambda_Function: Refreshing state... [id=AWS_S3_API_Lambda_Function]
aws_iam_role_policy.apigatway_policy: Refreshing state... [id=apigatway-role:apigatway-policy]
aws_api_gateway_resource.rest_api_resources: Refreshing state... [id=zgrj2z]
aws_api_gateway_method.rest_api_method1: Refreshing state... [id=agm-8ykbc3cw11-zgrj2z-GET]
aws_api_gateway_method_response.response_200: Refreshing state... [id=agmr-8ykbc3cw11-zgrj2z-GET-200]
aws_api_gateway_integration.rest_api_integration1: Refreshing state... [id=agi-8ykbc3cw11-zgrj2z-GET]
aws_lambda_permission.apigw_lambda: Refreshing state... [id=AllowExecutionFromAPIGateway]
aws_s3_bucket_acl.api_acl: Refreshing state... [id=us-east-1-lambdabucketapi,public-read-write]
aws_lambda_permission.allow_bucket: Refreshing state... [id=AllowS3Invoke]
aws_s3_object.object: Refreshing state... [id=sample.txt]
aws_s3_bucket_acl.state_bucket_acl: Refreshing state... [id=statefile-bucket,private]
aws_api_gateway_integration_response.rest_api_integration_response: Refreshing state... [id=agir-8ykbc3cw11-zgrj2z-GET-200]
aws_s3_bucket_notification.aws-lambda-trigger: Refreshing state... [id=us-east-1-lambdabucketapi]
aws_api_gateway_deployment.MyDemoDeployment: Refreshing state... [id=lghalz]

No changes. Your infrastructure matches the configuration.
```

4. CloudFront

I.Brief:

CloudFront is a fast content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to customers globally with low latency, high transfer speeds, all within a developer-friendly environment.

II.Code (cloudfront.tf, variable.tf, terraform.tfvars)

GitHub: https://github.com/KanchanSoni16/terraform-s3-lambda-apigatway.git

III.Implementation Steps:

Step:1

terraform init

```
PS C:\terraform\lambda> terraform init
Error opening log file: open C:\terraform\AWS_Infra\log: is a directory

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/archive from the dependency lock file
- Using previously-installed hashicorp/aws v4.30.0
- Using previously-installed hashicorp/archive v2.2.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Step:2

terraform plan:

```
PS C:\terraform\lambda> terraform plan
Error opening log file: open C:\terraform\AWS_Infra\log: is a directory
data.archive_file.lambda_api_function: Reading...
data.archive_file.lambda_api_function: Read complete after 0s [id=2a98f8091bfa71590365c920f7a87eb681fb4257]
aws_api_gateway_rest_api.rest_api: Refreshing state... [id=yzcxuqa10m]
aws_iam_role.apigatway_role: Refreshing state... [id=apigatway-role]
aws_dynamodb_table.state-locking: Refreshing state... [id=state-locking]
aws_iam_role.api_role: Refreshing state... [id=api_role]
aws_s3_bucket.api_bucket: Refreshing state... [id=us-east-1-lambdabucketapi]
aws_s3_bucket.statefile-bucket: Refreshing state... [id=statefile-bucket]
aws_cloudwatch_log_group.cloudwatch_logs: Refreshing state... [id=/aws/lambda/AWS_S3_API_Lambda_Function]
aws_api_gateway_resource.rest_api_resources: Refreshing state... [id=pzya8y]
aws_cloudfront_distribution.api_gateway_cf: Refreshing state... [id=E1K1ZN3LVIBU1Z]
aws_api_gateway_method.rest_api_method1: Refreshing state... [id=agm-yzcxuqa10m-pzya8y-GET]
aws_api_gateway_method_response.response_200: Refreshing state... [id=agmr-yzcxuqa10m-pzya8y-GET-200]
aws_lambda_function.AWS_S3_API_Lambda_Function: Refreshing state... [id=AWS_S3_API_Lambda_Function]
aws_iam_role_policy.api_policy: Refreshing state... [id=api_role:api_policy]
aws_iam_role_policy.apigatway_policy: Refreshing state... [id=apigatway-role:apigatway-policy]
aws_s3_bucket_acl.state_bucket_acl: Refreshing state... [id=statefile-bucket,private]
aws_s3_bucket_acl.api_acl: Refreshing state... [id=us-east-1-lambdabucketapi,public-read-write]
aws_s3_object.object: Refreshing state... [id=sample.txt]
aws_lambda_permission.apigw_lambda: Refreshing state... [id=AllowExecutionFromAPIGateway]
aws_api_gateway_integration.rest_api_integration1: Refreshing state... [id=agi-yzcxuqa10m-pzya8y-GET]
aws_lambda_permission.allow_bucket: Refreshing state... [id=AllowS3Invoke]
aws_s3_bucket_notification.aws-lambda-trigger: Refreshing state... [id=us-east-1-lambdabucketapi]
aws_api_gateway_deployment.apideploy: Refreshing state... [id=wd5vmf]
aws_api_gateway_integration_response.rest_api_integration_response: Refreshing state... [id=agir-yzcxuqa10m-pzya8y-GET-200]

No changes. Your infrastructure matches the configuration.
```

Step:3

terraform apply:

```
PS C:\terraform\lambda> terraform apply -auto-approve
Error opening log file: open C:\terraform\AWS_Infra\log: is a directory
data.archive_file.lambda_api_function: Reading...
data.archive_file.lambda_api_function: Read complete after 0s [id=2a98f8091bfa71590365c920f7a87eb681fb4257]
aws_s3_bucket.api_bucket: Refreshing state... [id=us-east-1-lambdabucketapi]
aws_dynamodb_table.state-locking: Refreshing state... [id=state-locking]
aws_api_gateway_rest_api.rest_api: Refreshing state... [id=yzcxuqa10m]
aws_cloudwatch_log_group.cloudwatch_logs: Refreshing state... [id=/aws/lambda/AWS_S3_API_Lambda_Function]
aws_s3_bucket.statefile-bucket: Refreshing state... [id=statefile-bucket]
aws_iam_role.api_role: Refreshing state... [id=api_role]
aws_iam_role.apigatway_role: Refreshing state... [id=apigatway-role]
aws_api_gateway_resource.rest_api_resources: Refreshing state... [id=pzya8y]
aws_cloudfront_distribution.api_gateway_cf: Refreshing state... [id=E1K1ZN3LVIBU1Z]
aws_api_gateway_method.rest_api_method1: Refreshing state... [id=agm-yzcxuqa10m-pzya8y-GET]
aws_api_gateway_method_response.response_200: Refreshing state... [id=agmr-yzcxuqa10m-pzya8y-GET-200]
aws_iam_role_policy.api_policy: Refreshing state... [id=api_role:api_policy]
aws_lambda_function.AWS_S3_API_Lambda_Function: Refreshing state... [id=AWS_S3_API_Lambda_Function]
aws_iam_role_policy.apigatway_policy: Refreshing state... [id=apigatway-role:apigatway-policy]
aws_api_gateway_integration.rest_api_integration1: Refreshing state... [id=agi-yzcxuqa10m-pzya8y-GET]
aws_lambda_permission.apigw_lambda: Refreshing state... [id=AllowExecutionFromAPIGateway]
aws_lambda_permission.allow_bucket: Refreshing state... [id=AllowS3Invoke]
aws_s3_bucket_acl.api_acl: Refreshing state... [id=us-east-1-lambdabucketapi,public-read-write]
aws_s3_object.object: Refreshing state... [id=sample.txt]
aws_s3_bucket_acl.state_bucket_acl: Refreshing state... [id=statefile-bucket,private]
aws_api_gateway_deployment.apideploy: Refreshing state... [id=wd5vmf]
aws_api_gateway_integration_response.rest_api_integration_response: Refreshing state... [id=agir-yzcxuqa10m-pzya8y-GET-200]
aws_s3_bucket_notification.aws-lambda-trigger: Refreshing state... [id=us-east-1-lambdabucketapi]

No changes. Your infrastructure matches the configuration.
```

5. Terraform Backend & Terraform Output

I. Brief:

A **backend** defines where terraform stores its state data files. Terraform Back can be anything like : AWS S3, Terraform Cloud , Google cloud etc.

Output values return information about our infrastructure on the command line and can give information for other Terraform configurations to use. Output values are like return values in programming languages.

II. Code (output.tf, backend-resources.tf, terraform.tf, variable.tf, terraform.tfvars)

GitHub: https://github.com/KanchanSoni16/terraform-s3-lambda-apigatway.git

III. Implementation Steps:

Step:1

terraform init

```
PS C:\terraform\lambda> terraform init
Error opening log file: open C:\terraform\AWS_Infra\log: is a directory

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/archive from the dependency lock file
- Using previously-installed hashicorp/aws v4.30.0
- Using previously-installed hashicorp/archive v2.2.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Step:2

terraform plan:

```
PS C:\terraform\lambda> terraform plan
Error opening log file: open C:\terraform\AWS_Infra\log: is a directory
data.archive_file.lambda_api_function: Reading...
data.archive_file.lambda_api_function: Read complete after 0s [id=2a98f8091bfa71590365c920f7a87eb681fb4257]
aws_api_gateway_rest_api.rest_api: Refreshing state... [id=yzcxuqa10m]
aws_iam_role.apigatway_role: Refreshing state... [id=apigatway-role]
aws_dynamodb_table.state-locking: Refreshing state... [id=state-locking]
aws_iam_role.api_role: Refreshing state... [id=api_role]
aws_s3_bucket.api_bucket: Refreshing state... [id=us-east-1-lambdabucketapi]
aws_s3_bucket.statefile-bucket: Refreshing state... [id=statefile-bucket]
aws_cloudwatch_log_group.cloudwatch_logs: Refreshing state... [id=/aws/lambda/AWS_S3_API_Lambda_Function]
aws_api_gateway_resource.rest_api_resources: Refreshing state... [id=pzya8y]
aws_cloudfront_distribution.api_gateway_cf: Refreshing state... [id=E1K1ZN3LVIBU1Z]
aws_api_gateway_method.rest_api_method1: Refreshing state... [id=agm-yzcxuqa10m-pzya8y-GET]
aws_api_gateway_method_response.response_200: Refreshing state... [id=agmr-yzcxuqa10m-pzya8y-GET-200]
aws_lambda_function.AWS_S3_API_Lambda_Function: Refreshing state... [id=AWS_S3_API_Lambda_Function]
aws_iam_role_policy.api_policy: Refreshing state... [id=api_role:api_policy]
aws_iam_role_policy.apigatway_policy: Refreshing state... [id=apigatway-role:apigatway-policy]
aws_s3_bucket_acl.state_bucket_acl: Refreshing state... [id=statefile-bucket,private]
aws_s3_bucket_acl.api_acl: Refreshing state... [id=us-east-1-lambdabucketapi,public-read-write]
aws_s3_object.object: Refreshing state... [id=sample.txt]
aws_lambda_permission.apigw_lambda: Refreshing state... [id=AllowExecutionFromAPIGateway]
aws_api_gateway_integration.rest_api_integration1: Refreshing state... [id=agi-yzcxuqa10m-pzya8y-GET]
aws_lambda_permission.allow_bucket: Refreshing state... [id=AllowS3Invoke]
aws_s3_bucket_notification.aws-lambda-trigger: Refreshing state... [id=us-east-1-lambdabucketapi]
aws_api_gateway_deployment.apideploy: Refreshing state... [id=wd5vmf]
aws_api_gateway_integration_response.rest_api_integration_response: Refreshing state... [id=agir-yzcxuqa10m-pzya8y-GET-200]

No changes. Your infrastructure matches the configuration.
```

Step:3

terraform apply:

```
PS C:\terraform\lambda> terraform apply -auto-approve
Error opening log file: open C:\terraform\AWS_Infra\log: is a directory
data.archive_file.lambda_api_function: Reading...
data.archive_file.lambda_api_function: Read complete after 0s [id=2a98f8091bfa71590365c920f7a87eb681fb4257]
aws_s3_bucket.api_bucket: Refreshing state... [id=us-east-1-lambdabucketapi]
aws_dynamodb_table.state-locking: Refreshing state... [id=state-locking]
aws_api_gateway_rest_api.rest_api: Refreshing state... [id=yzcxuqa10m]
aws_cloudwatch_log_group.cloudwatch_logs: Refreshing state... [id=/aws/lambda/AWS_S3_API_Lambda_Function]
aws_s3_bucket.statefile-bucket: Refreshing state... [id=statefile-bucket]
aws_iam_role.api_role: Refreshing state... [id=api_role]
aws_iam_role.apigatway_role: Refreshing state... [id=apigatway-role]
aws_api_gateway_resource.rest_api_resources: Refreshing state... [id=pzya8y]
aws_cloudfront_distribution.api_gateway_cf: Refreshing state... [id=E1K1ZN3LVIBU1Z]
aws_api_gateway_method.rest_api_method1: Refreshing state... [id=agm-yzcxuqa10m-pzya8y-GET]
aws_api_gateway_method_response.response_200: Refreshing state... [id=agmr-yzcxuqa10m-pzya8y-GET-200]
aws_iam_role_policy.api_policy: Refreshing state... [id=api_role:api_policy]
aws_lambda_function.AWS_S3_API_Lambda_Function: Refreshing state... [id=AWS_S3_API_Lambda_Function]
aws_iam_role_policy.apigatway_policy: Refreshing state... [id=apigatway-role:apigatway-policy]
aws_api_gateway_integration.rest_api_integration1: Refreshing state... [id=agi-yzcxuqa10m-pzya8y-GET]
aws_lambda_permission.apigw_lambda: Refreshing state... [id=AllowExecutionFromAPIGateway]
aws_lambda_permission.allow_bucket: Refreshing state... [id=AllowS3Invoke]
aws_s3_bucket_acl.api_acl: Refreshing state... [id=us-east-1-lambdabucketapi,public-read-write]
aws_s3_object.object: Refreshing state... [id=sample.txt]
aws_s3_bucket_acl.state_bucket_acl: Refreshing state... [id=statefile-bucket,private]
aws_api_gateway_deployment.apideploy: Refreshing state... [id=wd5vmf]
aws_api_gateway_integration_response.rest_api_integration_response: Refreshing state... [id=agir-yzcxuqa10m-pzya8y-GET-200]
aws_s3_bucket_notification.aws-lambda-trigger: Refreshing state... [id=us-east-1-lambdabucketapi]

No changes. Your infrastructure matches the configuration.
```

6. Git Push automation

I. Brief:

A **Git Push** defines to push the complete code automatically in git repo “awesome_project”.

II. Code (git.sh, local_provisioner.tf, terraform.tf, variable.tf, terraform.tfvars)

GitHub: https://github.com/KanchanSoni16/terraform-s3-lambda-apigatway.git

III. Implementation Steps:

Step:1

terraform init

```
PS C:\terraform\lambda> terraform init
Error opening log file: open C:\terraform\AWS_Infra\log: is a directory

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Reusing previous version of hashicorp/archive from the dependency lock file
- Using previously-installed hashicorp/aws v4.30.0
- Using previously-installed hashicorp/archive v2.2.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

Step:2

terraform plan:

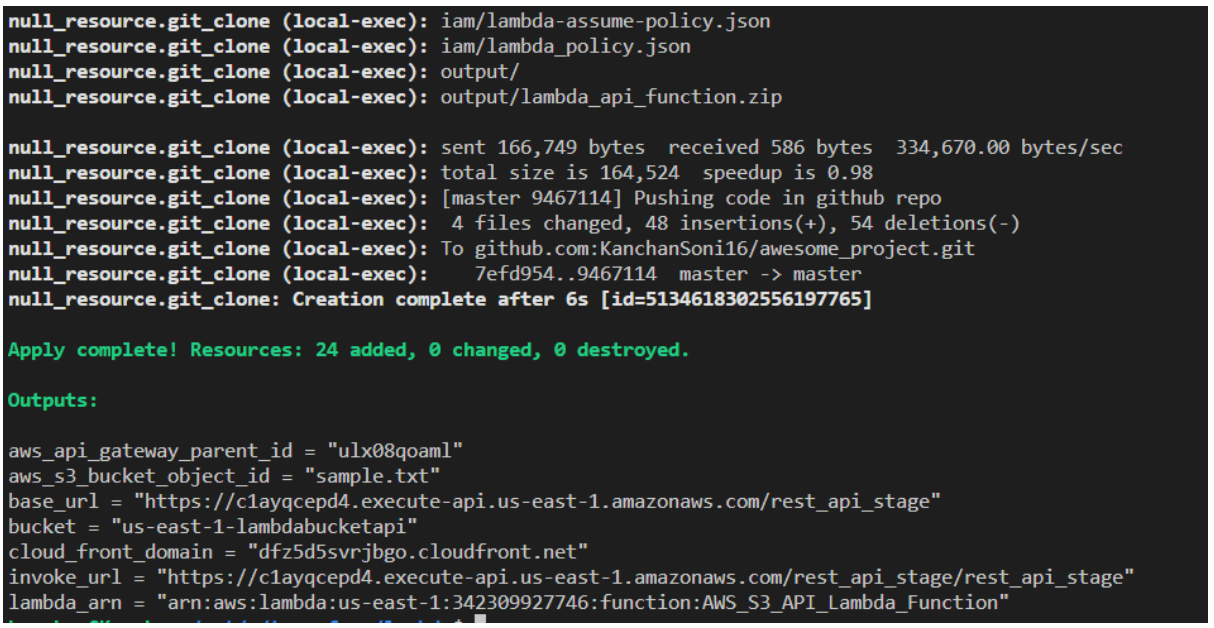
```
PS C:\terraform\lambda> terraform plan
Error opening log file: open C:\terraform\AWS_Infra\log: is a directory
data.archive_file.lambda_api_function: Reading...
data.archive_file.lambda_api_function: Read complete after 0s [id=2a98f8091bfa71590365c920f7a87eb681fb4257]
aws_api_gateway_rest_api.rest_api: Refreshing state... [id=yzcxuqa10m]
aws_iam_role.apigatway_role: Refreshing state... [id=apigatway-role]
aws_dynamodb_table.state-locking: Refreshing state... [id=state-locking]
aws_iam_role.api_role: Refreshing state... [id=api_role]
aws_s3_bucket.api_bucket: Refreshing state... [id=us-east-1-lambdabucketapi]
aws_s3_bucket.statefile-bucket: Refreshing state... [id=statefile-bucket]
aws_cloudwatch_log_group.cloudwatch_logs: Refreshing state... [id=/aws/lambda/AWS_S3_API_Lambda_Function]
aws_api_gateway_resource.rest_api_resources: Refreshing state... [id=pzya8y]
aws_cloudfront_distribution.api_gateway_cf: Refreshing state... [id=E1K1ZN3LVIBU1Z]
aws_api_gateway_method.rest_api_method1: Refreshing state... [id=agm-yzcxuqa10m-pzya8y-GET]
aws_api_gateway_method_response.response_200: Refreshing state... [id=agmr-yzcxuqa10m-pzya8y-GET-200]
aws_lambda_function.AWS_S3_API_Lambda_Function: Refreshing state... [id=AWS_S3_API_Lambda_Function]
aws_iam_role_policy.api_policy: Refreshing state... [id=api_role:api_policy]
aws_iam_role_policy.apigatway_policy: Refreshing state... [id=apigatway-role:apigatway-policy]
aws_s3_bucket_acl.state_bucket_acl: Refreshing state... [id=statefile-bucket,private]
aws_s3_bucket_acl.api_acl: Refreshing state... [id=us-east-1-lambdabucketapi,public-read-write]
aws_s3_object.object: Refreshing state... [id=sample.txt]
aws_lambda_permission.apigw_lambda: Refreshing state... [id=AllowExecutionFromAPIGateway]
aws_api_gateway_integration.rest_api_integration1: Refreshing state... [id=agi-yzcxuqa10m-pzya8y-GET]
aws_lambda_permission.allow_bucket: Refreshing state... [id=AllowS3Invoke]
aws_s3_bucket_notification.aws-lambda-trigger: Refreshing state... [id=us-east-1-lambdabucketapi]
aws_api_gateway_deployment.apideploy: Refreshing state... [id=wd5vmf]
aws_api_gateway_integration_response.rest_api_integration_response: Refreshing state... [id=agir-yzcxuqa10m-pzya8y-GET-200]

No changes. Your infrastructure matches the configuration.
```

Step:3

terraform apply

```
aws_cloudfront_distribution.api_gateway_cf: Provisioning with 'local-exec'...
aws_cloudfront_distribution.api_gateway_cf (local-exec): Executing: ["/bin/sh" "-c" "./graph.sh"]
aws_cloudfront_distribution.api_gateway_cf: Creation complete after 3m39s [id=EIC5CPJGQOUON]
null_resource.git_clone: Creating...
null_resource.git_clone: Provisioning with 'local-exec'...
null_resource.git_clone (local-exec): Executing: ["/bin/sh" "-c" "./git.sh"]
null_resource.git_clone (local-exec): Cloning into 'awesome_project'...
null_resource.git_clone (local-exec): Warning: Permanently added the ECDSA host key for IP address '140.82.114.3' to the list of known hosts.
null_resource.git_clone (local-exec): sending incremental file list
null_resource.git_clone (local-exec): ./
null_resource.git_clone (local-exec): .gitignore
null_resource.git_clone (local-exec): .terraform.lock.hcl
null_resource.git_clone (local-exec): .terraform.tfstate.lock.info
null_resource.git_clone (local-exec): api_gateway.tf
null_resource.git_clone (local-exec): apigatway-integration-role.tf
null_resource.git_clone (local-exec): aws_s3.tf
null_resource.git_clone (local-exec): backend-resources.tf
null_resource.git_clone (local-exec): cloudfront.tf
null_resource.git_clone (local-exec): git.sh
null_resource.git_clone (local-exec): graph.dot
null_resource.git_clone (local-exec): graph.sh
null_resource.git_clone (local-exec): graph.svg
null_resource.git_clone (local-exec): lambda-function.tf
null_resource.git_clone (local-exec): lambda-role.tf
null_resource.git_clone (local-exec): lambda_api_function.py
null_resource.git_clone (local-exec): local_provisioner.tf
null_resource.git_clone (local-exec): output.tf
null_resource.git_clone (local-exec): provider.tf
null_resource.git_clone (local-exec): sample.py
null_resource.git_clone (local-exec): terraform.tf
null_resource.git_clone (local-exec): terraform.tfstate
null_resource.git_clone (local-exec): terraform.tfvars
null_resource.git_clone (local-exec): variable.tf
null_resource.git_clone (local-exec): Object/
null_resource.git_clone (local-exec): Object/sample.txt
null_resource.git_clone (local-exec): iam/
null_resource.git_clone (local-exec): iam/apigateway-assume-policy.json
null_resource.git_clone (local-exec): iam/apigateway-policy.json
null_resource.git_clone (local-exec): iam/lambda-assume-policy.json
null_resource.git_clone (local-exec): iam/lambda_policy.json
```


null_resource.git_clone (local-exec): iam/lambda-assume-policy.json
null_resource.git_clone (local-exec): iam/lambda_policy.json
null_resource.git_clone (local-exec): output/
null_resource.git_clone (local-exec): output/lambda_api_function.zip
null_resource.git_clone (local-exec): sent 166,749 bytes received 586 bytes 334,670.00 bytes/sec
null_resource.git_clone (local-exec): total size is 164,524 speedup is 0.98
null_resource.git_clone (local-exec): [master 9467114] Pushing code in github repo
null_resource.git_clone (local-exec): 4 files changed, 48 insertions(+), 54 deletions(-)
null_resource.git_clone (local-exec): To github.com:KanchanSoni16/awesome_project.git
null_resource.git_clone (local-exec): 7efd954..9467114 master -> master
null_resource.git_clone: Creation complete after 6s [id=5134618302556197765]
Apply complete! Resources: 24 added, 0 changed, 0 destroyed.
Outputs:
aws_api_gateway_parent_id = "ulx08qoaml"
aws_s3_bucket_object_id = "sample.txt"
base_url = "https://c1ayqcepd4.execute-api.us-east-1.amazonaws.com/rest_api_stage"
bucket = "us-east-1-lambdabucketapi"
cloud_front_domain = "dfz5d5svrjbgo.cloudfront.net"
invoke_url = "https://c1ayqcepd4.execute-api.us-east-1.amazonaws.com/rest_api_stage/rest_api_stage"
lambda_arn = "arn:aws:lambda:us-east-1:342309927746:function:AWS_S3_API_Lambda_Function"

****